POLITECNICO
MILANO 1863

Computer Science and Engineering

PowerEnjoy Service - Project Plan

January 16, 2017

Prof. Luca Mottola

Authors:

- ZHOU YINAN(Mat. 872686)
- ZHAO KAIXIN(Mat. 875464)
- ZHAN YUAN(Mat. 806508)

Contents

1 Introduction

1.1 Revision History

Version 1.0

1.2 Purpose and Scope

This document aims at analyzing the overall complexity and making an estimation about the project size and required effort. The result will help project manager to decide the project budget, resource allocation and the schedule of activities.The document is divided into four parts.

In the first part of the document, We will use two specific methods to estimate the size and complexity of the project. First of all, we will use Function Points to calculate the average line of codes. Secondly, we will use COCOMO method to indicate the cost and effort estimation.

In the second part of the document, we will present the tasks for the project and the corresponding schedule. We will use the above results to come up with a suitable working plan covering the entire project development.

In the third part of the document, we will assign each team member specific missions to tickle down the project.

finally, we are going to analyze the risk we may encounter during the development. By analyzing the risk and coming up with possible solutions, we'll minimize the possibility for failure.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- Precedentedness: High if a product is similar to several previously developed projects
- Development Flexibility: High if there are no specific constraints to conform to pre-established requirements and external interface specs

- Architecture / Risk Resolution: High if we have a good risk management plan, clear definition of budget and schedule, focus on architectural definition
- Team Cohesion: High if all stakeholders are able to work in a team and share the same vision and commitment.
- Process Maturity: Refers to a well known method for assessing the maturity of a software organization, CMM, now evolved into CMMI

1.3.2 Acronyms , Abbreviations

- FP : Function Point
- ILF : Internal Logic File
- ELF : External Logic File
- EI : External Input
- EO : External Output
- EQ : External Inquires
- PREC : Precedentedness
- FLEX : Development Flexibility
- RESL : Risk Resolution
- TEAM : Team Cohesion
- PMAT : Process Maturity

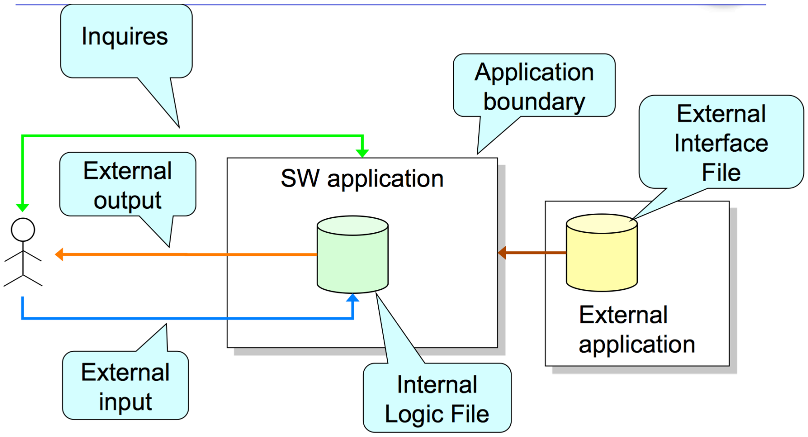
1.4 Reference Documents

- Specification Document Assignments AA 2016-2017
- Function Point tables
- COCOMO tables

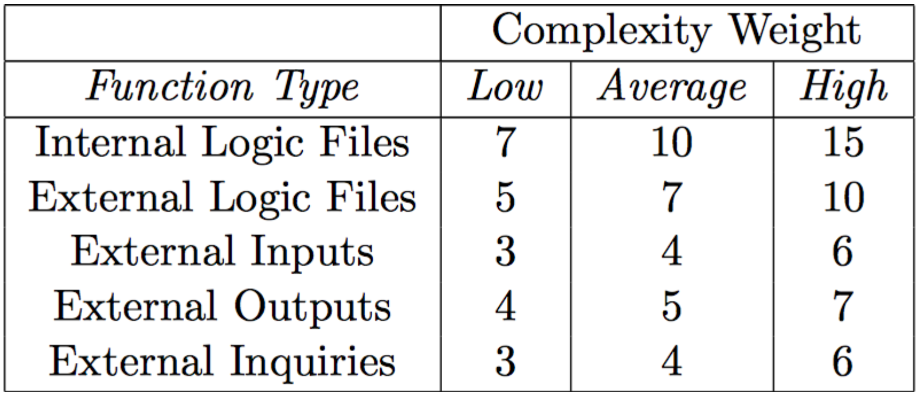
2 Project size, cost and effort estimation

We use functional points to estimate the software size. A Function Point (FP) is a unit of measurement to express the amount of business functionality, an information system (as a product) provides to a user. FPs measure software size. They are widely accepted as an industry standard for functional sizing.
Functional Points are based on a combination of program characteristics, more specifically :

- Data structures
- Inputs and outputs
- Inquires
- External interfaces



A weight is associated with each of these FP counts; the total is computed by multiplying each raw count by the weight and summing all partial values. The weight table for different Function types is described below.



	Complexity Weight		
Function Type	*Low*	*Average*	*High*
Internal Logic Files	7	10	15
External Logic Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

2.1 Size estimation: function points

2.1.1 Internal Logic Files(ILFs)

Internal Logical File (ILF): homogeneous set of data used and managed by the application. In our application, there are a few tables we need to manage in the ILFs in order to provide functional requirements.

- User table : The User table maintains all the valid registered users, including the credentials and paymentInfo. More specifically, the credential must include name, email address, login code and number of driving license. Payment information includes number of bank account or credit card, expired time, security code, holder name and Phone number.

- Car table : The table maintains all the information about the cars, including car plate, capacity, current state and onCarDev info.

- Reservation table : The Reservation table maintains all the cars that are currently reserved. The table has a map information between user(email) and car(plate). Also the table maintains the remaining time left for the user to pick up the car.

- Ride table : The Ride table maintains all the cars that are currently under working. The table has a map between user(email) and car(plate).

- DiscountAndPunish table : Whenever a discount or punishment happens, we store the corresponding information in it. The table has the user email, car plate, the discount or punishment amount and a short description.

- Area table : the Area table stores all the safe area information.

Table 1: Internal Logic Files

	Data elements		
Record Element	1-19	20-50	51+
1	Low	Low	Avg
2-5	Low	Avg	High
6+	Avg	High	High

With the Internal Logic Files table above, we can estimate the FP for ILF.

Table 2: ILF

ILF	Complexity	FPs
User	HIGH	15
Car	HIGH	15
Ride	AVG	10
Reservation	AVG	10
DiscountAndPunishment	HIGH	15
TOTAL		65

2.1.2 External Logic Files(ELFs)

The external Logic Files in our application are data source from Google Map and bank services. For the bank services, our system just generate a request for remote bank service and then leave the user to interact with it. So we do not have much to do with the bank services. The Google Map external service, however, requires some modifications about the data we get.

Table 3: External Logic Files

	Data elements		
Record Element	1-19	20-50	51+
1	Low	Low	Avg
2-5	Low	Avg	High
6+	Avg	High	High

Table 4: ELF

ELF	Complexity	FPs
Bank Service	Low	5
Google Map	Avg	7
Total		12

2.1.3 External Inputs(EIs)

Our service deals with several situations which require user inputs.

- Register : Register operations is simple, just needs the client to fill a form and send to the server. The server will then check the validity of the info. For a positive result, a password is generated and send back to the user. The new user info will be inserted into the data base. Thus the complexity for it is low.
- Login : It is simple operation, just needs the server to check the validity of the email and password. The complexity for it is low.
- Get Available Cars : It is a bit complex operation. The user needs to send to the server the request. The server will need to analyze the location, look for available cars near the location. In order for this operation to proceed, several steps of interaction with different components is required. The complexity is high.
- Reserve : It is a simple operation. The user sends the request to server with the information about the reservation choice. The complexity is low.
- Unlock the Door : It is a simple operation. The user sends the request to the server with the info of his/her location information. The Complexity is low.
- Start Ride : It is a simple operation, only needs to send the request to the server and update the corresponding table. The complexity is low.
- End Ride : Unlike Start Ride, End Ride is a bit more complex. Besides sending the request to the server, the system also needs to compute the total cost, update the corresponding table, and call external bank service. The complexity for this operation is high.

- Modify info : This operation includes modification of existing information in the system. It is a simple operation, since it just needs to check the validity and update the corresponding table. The complexity is low. The average atomic modification includes inserting, updating and deleting. By matching all of them to User, Car and SafeArea. The overall is 3*10 = 30 atomic modifications.

Table 5: External input

	Data Elements		
File Types	1-5	5-15	16+
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Table 6: EI

EI	Complexity	FPs
Register	Low	3
Log in	Low	3
Get Available Cars	High	6
Reserve	Low	3
Unlock Door	Low	3
Start Ride	Low	3
End Ride	High	6
Modification	Low	3*10
Total		60

2.1.4 External Inquires(EQs)

External quires allow users to retrieve information from the server.

- User retrieve his/her credentials/PaymentInfo : It is a simple operation because it only needs to find info from data base and send back to the user. The complexity is low.

- Get safeAreas : It is a simple operation which only deals with the data base. The complexity is low.

- Get DiscountsAndPunishment : It is a simple operation which only deals with the data base. The complexity is low.

Table 7: External Query

	Data Elements		
File Types	1-5	5-15	16+
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Table 8: EQ

EQ	Compexity	FPs
Retrive Info	Low	3
Get SafeAreas	Low	3
Get Discounts and Punishment	Low	3
Total		9

2.1.5 External Outputs(EOs)

The system needs to notify the user and the car the corresponding information.

- Notify the user expired Reservation time
- Notify the user completion of the payment.
- Notify the user the execution of punishment.
- Notify the car the current Location and price.

All the operations are fairly simple, except for the last one which needs to interact with several IFLs and EFLs.

Table 9: External Output

	Data Elements		
File Types	1-5	5-15	16+
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Table 10: EO

EO	Compexity	FPs
Notify expired reservation time	Low	4
Notify the completion of payment	Low	4
Notify the execution of punishment	Low	4
Notify the car Location and Price	High	6
Total		18

2.1.6 Overal estinamtion

The following table summarize the overall estimation.

Table 11: Overall estimation

Function Type	Value
Internal Logic File	65
External Logic File	12
External Input	60
External Query	9
External Output	18
Total	164

Consider using JEE as the development tool, and the additional work for User and Cap Applicationm, we get the following estimated lines of code:

$$\boxed{\text{SLOC} = 164 * 67 = 12464}$$

2.2 Cost and effort estimation: COCOMO II

2.2.1 Scale Drivers

2.2.2 Cost Drivers

2.2.3 Effort equations

2.2.4 Schedule estimation

3 Schedule

4 Resource allocation

5 Risk management

In this section, we'll analyze the risk we may encounter during the project development. Basically, there are three kinds of risks : Project risks, Technical risks, and Business risks. We'll present specific risks we may be trapped with and what we can do to deal with it. We use [L,M,H] to indicate the probability the risk will occur. [Low,Moderate,High]

The most possible risks come from the technical part. They threaten the quality and timeliness of the software to be produced. Some possible risks may be :

- Wrong functionality [M]: Wrong functional quality can be a serious risk and lead to meaningless workload which increases the cost and slow down the project.
  To deal with wrong functionality, we adopt a proactive risk strategy. By better writing the RASD document and increase the meeting frequency with the stakeholders, we avoid this kind of risk. Also frequently report the project process and get feedback from the stakeholders will help.

- Wrong User Interface[M] : Wrong User Interface can lead to meaningless workload.
  To deal with it, we need to frequently meet with the stakeholders and make necessary discussion.

- Bad external components[L] : Bad external components are a major issue and threat to our project. Our project largely depend on the reliability of the following external components : Google Map, DBMS, Bank Service. In case any of them fail, we need to rewrite the business logic components and this leads to significant workload and time increase.

  Although the consequence is serious, the possibility of this risk to happen is actually low. Since the external components we choose to use is from huge and reliable companies like Google and Oracle. So we adopt a reactive risk strategy here.

- Fail to accomplish logic components[L] : This risk is purely technical and there are no good solutions. Either we give programmers time to tick the task down or we recruit new employers with experience. Since the actual problem can be various, it is difficult to come up with a specific prediction. Here we adopt a reactive strategy.

The project risk may arise during the software development. These risks mainly come from the stakeholder side.

- Requirements volatility[M] : Requirement may change during the development. The point is we can not add too much constraints on our stakeholders. Here we adopt a proactive approach by organizing frequently meetings with the stakeholders. By doing this, we may avoid the risk and minimize the negative consequence.
- Unrealistic schedule/budget[M] : This risk comes from both our own side and stakeholder side. During the development we may encounter various uncertain situations and risks which will slow down our process. Also budget may be tight. Here we adopt a reactive approach because we do not know how to come up with a efficient solution until we actually get trapped by the problem.

The third part of the risk is business risk. Some possible situations are presented below:

- Management risk[L] : There is a possibility of losing the support of senior management due to a change in focus or a change in people. Since we assume there are three members in our development team, it is actually kind of critical issue, however the possibility is not high. So we adopt a reactive strategy here.
- Budget risk[L] : Losing budget or personnel commitment is not so likely in our case, so we adopt a reactive strategy here.

6 Effort